Chapter 6
Efficient and Deadlock-Free Tree-Based Multicast Routing Methods for Networks-on-Chip (NoC)

Faizal Arya Samman and Thomas Hollstein

Abstract This chapter presents a new efficient and deadlock free tree-based multicast routing method and concept. The presented deadlock-free multicast routing algorithm can be implemented on a network-on-chip (NoC) router microarchitecture, realizing a mesh planar network topology. The NoC microarchitecture supports both deadlock-free static and efficient adaptive tree-based multicast routing. Multicast packets are routed and scheduled in the NoC by using a flexible multiplexing/interleaving technique with wormhole switching. The flexibility of the proposed multicast routing method is based on a locally managed packet identity (ID-tag) attached to every flit. This concept allows different packets to be interleaved at flit-level in a single buffer pool on the same link. Furthermore, a pheromone tracking strategy presented in this chapter, which is used to reduce communication energy in the adaptive tree-based multicast routing method. The strategy is used to perform efficient spanning trees for the adaptive tree-based multicast routing which are generated at runtime.

6.1 Introduction

Recent multicomputers and large-scale multiprocessor systems have been developed towards collective communication services to reduce communication overheads of parallel computations running on these systems. These collective communication

F.A. Samman (✉)
Universitas Hasanuddin, Fakultas Teknik, Jurusan Teknik Elektro,
Jl. Perintis Kemerdekaan Km. 10, Makassar 90245, Indonesia
e-mail: faizalas@unhas.ac.id

T. Hollstein
Department of Computer Engineering, Tallinn University of Technology,
Akadeemia tee 15A, 12618 Tallinn, Estonia
e-mail: thomas@ati.ttu.ee

M. Palesi and M. Daneshtalab (eds.), *Routing Algorithms in Networks-on-Chip*,
DOI 10.1007/978-1-4614-8274-1_6,

services include *one-to-many* communication such as *multicast* (the same message is sent from one source node to an arbitrary number of destination nodes), *one-to-all* communication such as *broadcast* (the same message is sent from one source node to all nodes (entries) in the network) and *scatter* (different messages are sent from one source node to all entries in the network), *many-to-one* communication (one destination node receives different messages from an arbitrary number of source nodes), and *all-to-one* communication such as *reduce* (one destination node combines different messages from an arbitrary number of source nodes by performing a certain operation such addition, multiplication, maximum, minimum, or a logical operation).

Among the aforementioned collective communications, multicast and broadcast communications are the most interesting communication modes. Both communication modes are needed in many parallel algorithms and applications in multiprocessor systems. They also require special attention and handling in the network communication protocol layers. A processing node in a multiprocessor system can inject a multicast message into a network by sending separate individual copies of the message from the source to every destination node. However, this unicast-based multicast delivery is energy- and time-consuming [24].

In large-scale off-chip multiprocessor systems, collective communication services, such as multicast communication, have been a fundamental prerequisite for some data parallel computer languages. In a distributed shared-memory (DSM) parallel programming model, the multicast data communication service is applied to efficiently support shared-data invalidation and updating on distributed nodes. This is being used in numerous parallel algorithms, e.g. parallel search and parallel graph algorithms. In a single-program multiple-data (SPMD) and in a data parallel programming models, a variety of process control operations and global data movements such as reduction, replication, permutation, segmented scan and barrier synchronization [16] can benefit from multicast services.

In the Multiprocessor System-on-Chip (MPSoC) or multicores systems domains, parallel computing problems can potentially be solved based on parallel programming models. Therefore, multicast communication services, which should be implemented in upper protocol layers (typically on software level) and bottom protocol layers (typically on hardware level), will also be an important issue in a NoC-based multiprocessor context. Modern 3D chip stacking technologies enable on-Chip DSM systems with local DRAM memory being available at every processing node. Multicast communication services are in this context essential for efficient implementation of memory coherency protocols. Furthermore, in the upper protocol layers, multicast services are implemented as Application Programming Interface (API) routines (programming model) that can be used by users to develop parallel computing programs.

Further information on multicast routing issues can be found in [10]. This chapter is addressing a router hardware architecture supporting an efficient tree-based adaptive multicast routing method for NoCs that is implemented in bottom communication network protocol layers, i.e. on network, data link and physical layers according to the Open Systems Interconnection (OSI) model [30].

6.2 Occurrence of a Deadlock Problem Due to Multicast Dependency

This chapter describes the use of an efficient multicast routing method for NoCs. However, the use of the tree-based multicast routing method may lead to a multicast dependency problem ending up in a deadlock, as presented in Fig. 6.1. This multicast dependency can cause a multicast deadlock configuration (as described in Duato's Book [10]). In this case, multicast packets block each other and cannot move further.

The deadlock problem occurs especially if packets switched with the wormhole method or virtual cut-through switching are not short enough, there is not enough buffer space to store the contenting wormhole packet, and/or arbitration rules are not well organized to handle the multicast contention. In node (2,2) as presented in the figure, packet A blocks the flow of two multicast branches of packet B (to west and east), while in node (2,1), packet B blocks the flow of two multicast branches of packet A (to west and to east). Due to the "wait and hold" situation in both network switch nodes, both message A and B cannot move further.

In this chapter a NoC architecture called XHiNoC (eXtendable Hierarchical Network-on-Chip) is presented, which proposes a *novel wormhole cut-through switching* concept [26] based on a local identity-based (ID-based) interleaved routing organization, in which the ID-tag of each packet is locally updated on each communication link [23]. The XHiNoC's concept has also exhibited a novel multicast routing method [22, 24, 25] for NoCs based on the interleaved routing organization with local Identity (ID) management. In the XHiNoC routers, the just described deadlock configuration problem (due to multicast dependencies) is solved efficiently by using a so-called "*hold and release tagging mechanism*".

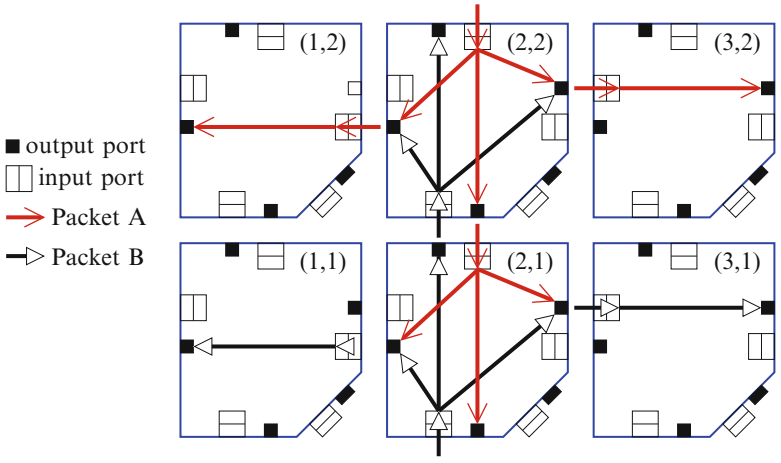

Fig. 6.1 Deadlock configuration in tree-based multicast routing

The hold-release tagging mechanism is based on an asynchronous data switching approach. A multicast flit will not be switched to an outgoing port, before its multicast request has been granted by the arbiter unit at the considered outgoing port. Each individual request can be granted asynchronously. Once the individual request is granted, it will be switched out towards the next link. The Hold-Release Tagging Mechanism applies the following principle to escape from flit contention due to multicast dependency. "*If a multicast flit from an input port n has an number N of requests at time t_s, then each single request to an output port m can be forwarded from the input port n to this output port m in the next time stage if and only if it receives a grant by an arbitration unit at the input port n, while the other requests must be held in the input port if they did not receive a grant by their requested output ports. In each next time stage, a single request, which has been granted before, must be reset to prevent improper flit replication. If all requests have been granted, then the multicast flit can be released from the queue in input port n* [25]".

Most of this chapter contents are adopted from our paper published by Elsevier Science [27]. Therefore, we would like to thank Elsevier Science for the Courtesy/Permission to republish some parts of the issue.

6.3 ID-Based Routing Organization

In the XHiNoC routing concept, an ID-tag is attached on each flit of a data packet. Flits belonging to the same data packet will have the same local ID-tag on each communication link. Based on this concept, multicast communications can be handled dynamically and pre-processing algorithms are not required before injecting multicast messages.

6.3.1 ID-Based Multiple Access (IDMA) Packet Stream Interleaving

Figure 6.2 illustrates, how five packets or data streams (*a*,*b*,*c*,*d* and *e*) can be interleaved on each communication link. For instance, the ID-tag allocation of the stream *a* sent from *Sw*4 to *Sw*3 via *Sw*5 and *Sw*2 can be seen. Its ID-tags are mapped and allocated to ID slot 2 on Local input link of *Sw*4, ID slot 1 on West input link of *Sw*5, ID slot 1 on North1 input link of *Sw*2, ID slot 1 on West input link of *Sw*3 and at last, to ID slot 0 on Local output link of *Sw*3. Flits belonging to the same packet or stream will have the same ID-tag on a specific link. Therefore, the ID-tag attached on every flit enables each packet or streaming data flit to be switched to the correct routed direction. In other words, the ID-tag represents the compressed form of the routing direction made by the header flit when reserving communication resources.

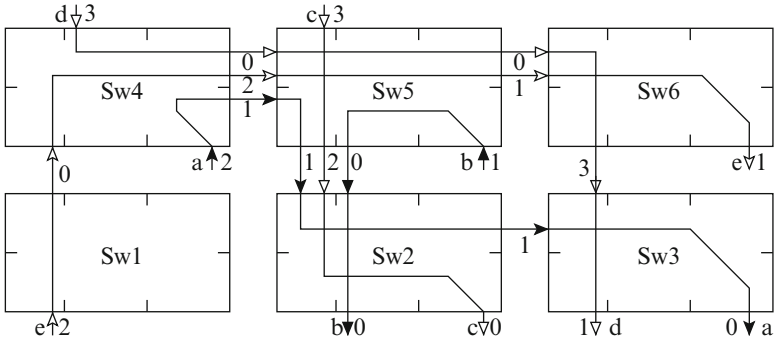

Fig. 6.2 ID-tag-based routing/switching organization and connection scheduling

Each streaming data flit can extract the required routing direction from the routing table that has been indexed in accordance with the local ID-tag of the packet stream.

6.3.2 Routing Table and ID Management

The local ID-tag attached on each flit of the packet (see Fig. 6.5) is updated and dynamically changed once a flit is switched to a new outgoing port. The ID update is made by an ID management unit located at every output port. Figure 6.2 shows the detailed view of the link sharing between stream *a*, *b* and *c* as presented in Fig. 6.2. The communication resource (link) connecting South1 output port of Switch 5 (*Sw*5) and North1 input port of Switch 2 (*Sw*2) assigns packet stream *a*, *b* and *c* with ID-tag 1, 0 and 2, respectively. Tables presented on the right side of the switches are the ID-slot table of the South1 output port at *Sw*5 and the routing table (*LUT*) of the North1 input port at *Sw*2. The content of the ID-slot table represents the ID-tag mapping function of each packet stream. The content of the routing table represents the routing directions to keep the correct routing tracks for each flit of the interleaved packet streams.

Figure 6.2 shows how the ID-tag of a stream header coming from NORTH port with ID-tag 3 is updated after being switched. The ID update process working as follows: When the IDM detects a new incoming stream or packet header, then it searches for a free ID slot on the output link by checking the ID-slot state in the corresponding table. In the example case, the ID-tag 2 is free. The ID is then assigned as the new ID-tag on the next link segment. The ID-slot 2 is indexed based on the associated incoming local ID-tag 3 and the incoming direction (NORTH). Hence, a data flit following the packet/stream at any instant time coming from NORTH port with ID-tag 3 will be assigned also with the new ID-tag 2 in the outgoing SOUTH port. In the same phase, the ID-tag 2 flag is set from “free” to “used” state, and the number of used IDs (UID) is incremented. When the UID has

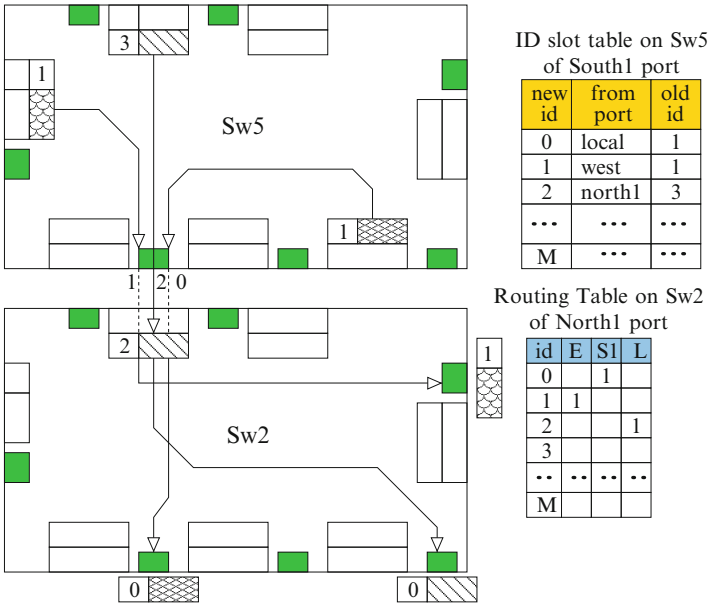

Fig. 6.3 IDMA: packet flit (wire-through-share) interleaving

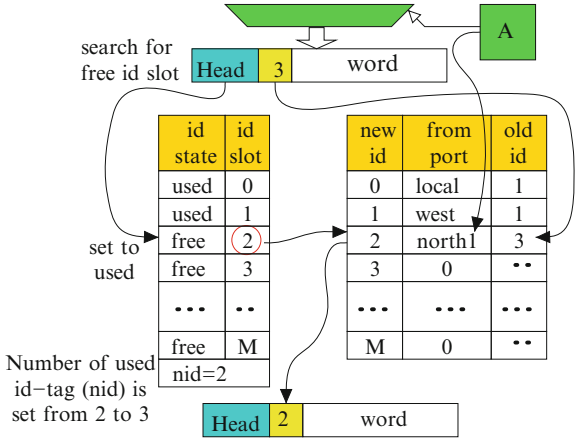

Fig. 6.4 IDMA: ID-tag updating

reached the number N of available ID slots, then "empty free ID flag" is set. When a tail flit (the end of stream data) is passing through the outgoing port, then the state of the related ID-tag 2 state is set from "used" to "free", the UID is decremented and the information related to this ID-number is then deleted from the ID Slot Table concurrently (Figs. 6.3 and 6.4).

More information about the dynamic ID management can be found in detail in [22–24, 26].

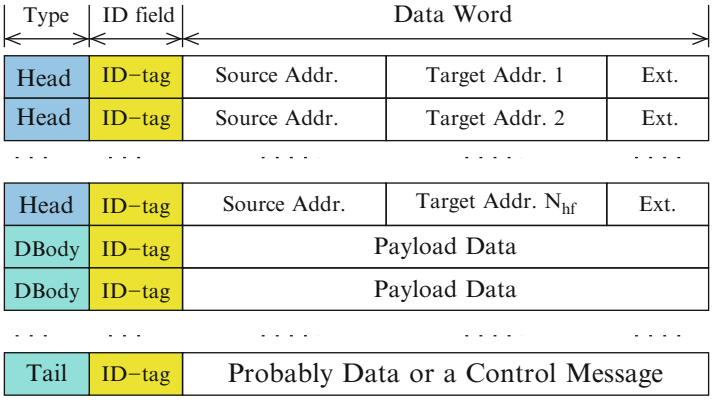

Fig. 6.5 The packet format for a multicast message

6.3.3 *Multicast Packet Format*

The packet format used in the XHiNoC, supporting the ID-based routing organization, is depicted in Fig. 6.5. A multicast packet consists of a number N_{hf} of header flits (which is equal to the number of multicast destinations N_{dest}), any amount of payload flits and a tail flit. The flit flow control field of every data flit consists of 3-bit flit *type* header and a 4-bit packet ID ($Identity$). Therefore in a 32-Bit instance of the system, each flit of the packet has 39-bit width, i.e. 32-bit data word plus 7-bit control field. The *type* can be *header*, *data body*, and the *end of databody* (*last/tail flit*) as shown in the Fig. 6.5. Flits belonging to the same message will always have the same local ID label on one and the same communication link. The ID number attached in each flit will vary over different communication links to support a *wire-sharing concept* with *flit-level message interleaving*. This concept scales well with increasing mesh network sizes.

Definition 1. A multicast message (even if the size is very large) is not divided into several sub-packets. Therefore, when N_{pf} number of payload flits that will be sent to N_{dest} number of destination nodes, the size of the multicast message will be $N_{hf} + N_{pf} + 1\ tail flit$.

6.4 Microarchitecture of the XHiNoC Router

6.4.1 *2D Mesh Planar Topology*

By for instance using the West-First routing algorithm, the adaptivity of the multicast-tree in the West direction is limited by the South–West and North–West prohibited turns, where multicast adaptive tree-based routing cannot be made if the

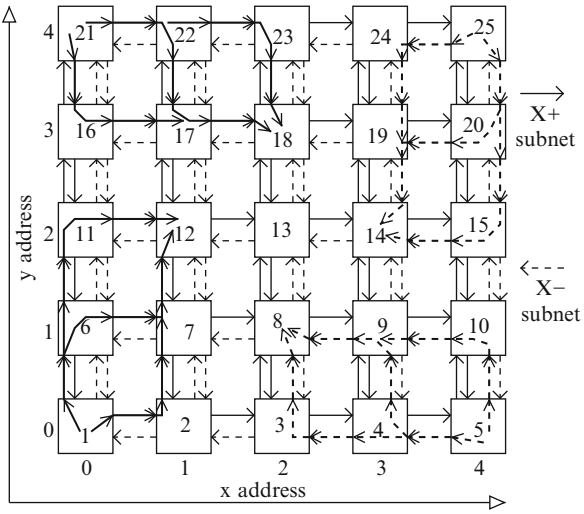

Fig. 6.6 2D planar mesh network

destination addresses are located in the South-West and North-West quadrant area [24]. These prohibited turns must be implemented in the routing algorithms to avoid the occurrence of a deadlock configuration. In order to cover such problems, a planar 2D NoC architecture with mesh topology is also presented in this chapter. The NoC is divided into two sub-networks in order to increase the degree of adaptivity of the routing functionality. A planar adaptive routing algorithm has been firstly introduced in [6], in which virtual channels (VCs) are introduced to support adaptive routing and to couple the sub networks. The main difference of the presented approach is that, instead of using VCs, we replace them with a double physical communication link to increase the link and switch bandwidth capacity.

Figure 6.6 shows an example of the 2-D mesh 5×5 network. The Network-on-Chip is physically divided into two subnetworks i.e., $X+$(depicted in solid line arrows) and $X-$subnetworks (depicted in dashed line arrows). If the x-distance between source and target nodes ($x_{offs} = x_{target} - x_{source}$) is zero or positive, then packets will be routed using the $X+$subnetwork. If x_{offs} is zero or negative, then the packets will be routed through the physical channels of the $X-$subnetwork. The ports connected with vertical y-direction links of $X+$ and $X-$subnetworks are denoted by (North1, South1) and (North2, South2), respectively. The packets being routed through the $X+$subnetwork will have adaptivity to make West–North1, West–South1, North1–East and South1–East turns as well as West–East, North1–South1 and South1–North1 straightforward (non-turn) routing direction. The packets being routed through the $X-$subnetwork will have adaptivity to make East–North2, East–South2, North2–West and South2–West turns as well as East–West, North2–South2 and South2–North2 straightforward routing directions.

The planar adaptive routing technique on a mesh topology has been firstly introduced in [6] and is deadlock-free by principle. Instead of using virtual channels to implement the interconnects between NORTH and SOUTH port as made in [6],

we prefer to implement two physical channels to separate the NORTH–SOUTH link interconnects for the $X+$ and $X-$subnetworks. The objectives for this modified topology are the preservation of the router performance and the increase of the network bandwidth. In case that virtual channels would have been implemented between the NORTH and SOUTH ports, then we needed to add two virtual queues at both incoming and outgoing ports. This would degrade the router performance characteristic or increase the data transfer latency, therefore we substitute VCs by adding two additional ports (NORTH2 and SOUTH2 ports) to the existing mesh router.

In a typical 5-port mesh switch, a 2D $N \times M$ mesh network will have $N \times (M-1) + M \times (N-1)$ full-duplex directional communication links. By using 2D planar-based mesh router, the $N \times M$ mesh network will provide more communication resources, i.e. $2N \times (M-1) + M \times (N-1)$ full-duplex directional links (and additionally the local core connections).

6.4.2 Static and 2D Planar Adaptive Routing Algorithms

In this chapter, we will evaluate different models of NoCs with mesh topology by using four mesh prototypes with different routing algorithms. The first model ('plnr' prototype) uses a 2D multicast planar adaptive routing algorithm that is presented in Algorithm 4 and is implemented based on the planar mesh topology. The adaptive routing decision is made based on the number of used ID slots (or the number of free ID slots) on each possible alternative routing direction and the record of the routing path made by other headers of the same message, which is later explained in Sect. 6.5.3 and presented in Algorithm 2. The routing algorithm is divided into two subalgorithms for $X+$ and $X-$subnetworks.

The second prototype ('xy' prototype) uses a static XY routing algorithm in the standard mesh topology, where messages are first routed in horizontal X-direction and then to vertical Y-direction. Hence, North–East turn and North–West as well as South–East and South–West are prohibited in the static XY routing algorithm. The remaining two models ('wf-v1' and 'wf-v2' prototypes) use the minimal West-First routing algorithms. In the West-First routing algorithm, packets will always be firstly routed to West direction, if its destination nodes are located in western area relative from its source node or current position. When the destination nodes are located in eastern area of its source node or current position, then the packet can be routed adaptively to East, North or South directions [10].

6.4.3 Generic Modules and Modular Design

The XHiNoC router microarchitecture is developed based on a modular concept and units are grouped into incoming block and outgoing block components as presented

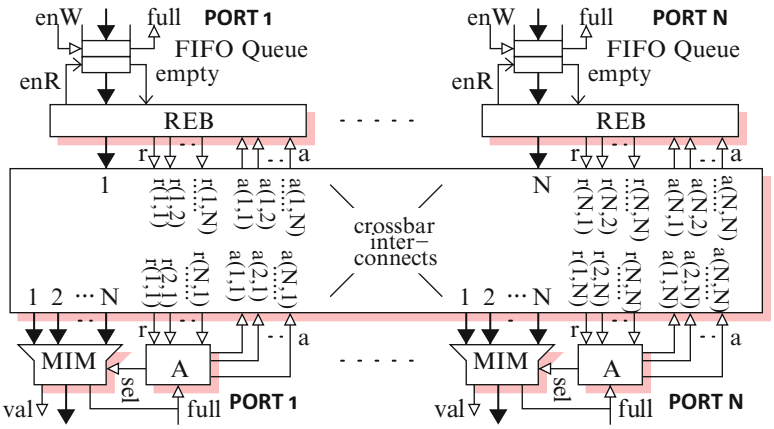

Fig. 6.7 General XHiNoC router architecture

in Fig. 6.7. Each module is modeled based on generic code, which is strongly related to the number of input-output connectivities of each port. The components located at input ports are *FIFO buffers* and a *Routing Engine with Data Buffering (REB)*. The components located at output ports are an *Arbiter (A)* and a *Crossbar Multiplexer with ID-Management Unit (MIM)*. The working principles and mechanisms of the Arbiter, REB and MIM units are explained in detail in [22] and [24].

Figure 6.8 shows the components in an incoming port of the XHiNoC router. In the REB module, there are a *Grant-Multicasting Controller* (GMC), a *Routing Engine* (RE), a *Route Buffer* and a *Read-Logic Unit* (RLU). The GMC consists of combinatorial logic and is used to control the acceptance of the multicast routing acknowledge (grant) signals from output ports. The RLU is also a combinatorial logic, which is used to control the read-operation of a flit from the FIFO buffer into the Route Buffer. When a routing direction for a flit is being decided by the RE unit, then this flit will be concurrently buffered in the Route Buffer. This concurrent step is introduced in order to reduce the number of internal pipeline stages in the XHiNoC router, and it can improve the router performance accordingly. The RE unit consists of the *Routing State Machine (RSM)* and the *Routing Reservation Table (RRT)*, which consists of a number of H preservable routing slots.

The design of the XHiNoC routers can be fully parametrized and customized on demand. Each VHDL entity contains generic code, which enables the derivation of new VHDL modules with a specific architecture and a number of input/output pins according to the specification. The custom-generic modular-based design approach also enables us to easily generate irregular NoC topologies.

Before running a real experiment, the different routing paths that will be performed by the aforementioned tree-based multicast routing methods (except for the wf-v1 multicast method) are shown in Fig. 6.9. A multicast message is injected from node (2,2) to 10 multicast destinations. The ‘xy’ multicast router performs 24 link usages in the NoC. While the ‘plnr’ and ‘wf-v2’ multicast routers perform

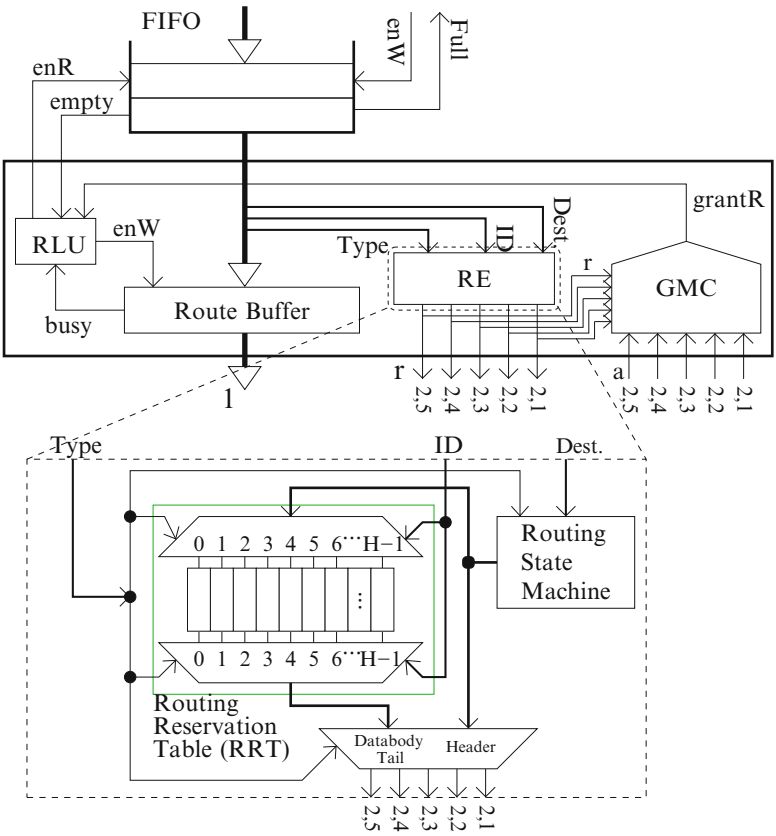

Fig. 6.8 Input port of a 5-port router (for the static XY routing, the signal paths of the number of used ID slots are removed)

with only 19 and 21 link segment usages in the NoC respectively. The traffic metric is defined as the number communication links used by a multicast packet to travel from a source node to destination nodes. From this example it can be concluded, that the planar adaptive multicast router can potentially reduces the communication energy of the multicast data transmission. The following experiment will show us the result of a more complex data distribution scenario.

6.5 Efficient Runtime Spanning Tree Configurations

The tree-based multicast routing algorithm implemented in the XHiNoC architecture is deadlock-free. The multicast NoC router architecture is designed and implemented based on the novel theory for deadlock-free multicast routing presented in [25].

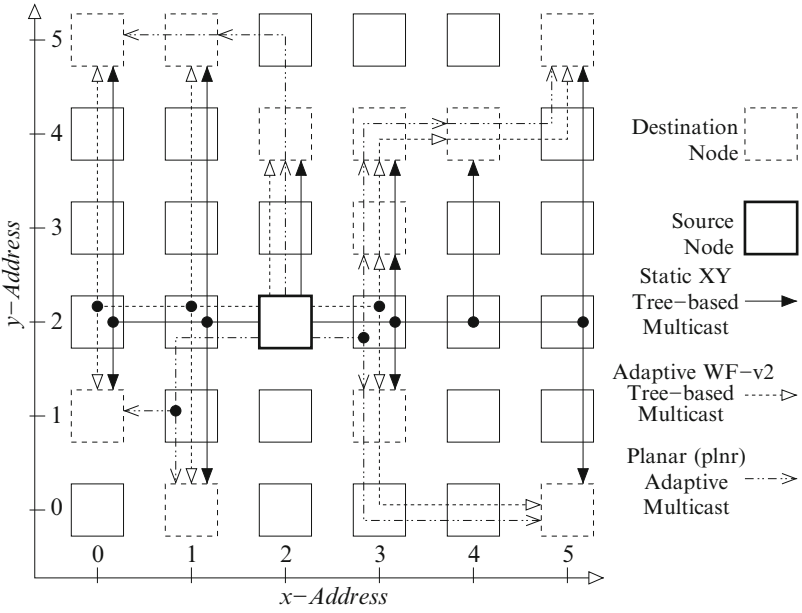

Fig. 6.9 The traffic patterns by using static tree-based, minimal adaptive west-first and minimal planar adaptive multicast routing methods

The problem of inefficient runtime spanning tree configuration that can affect the overall throughput of the generated multicast trees [24]. Because of this uncovered issue, in any circumstance, the adaptive routing cannot show better performance, since the data rate of the multicast tree depends on the slowest data rate in all spanning trees or branches of the multicast tree. Therefore, based on the presented local ID management concept, an *efficient method for runtime multicast spanning tree configurations* by using a minimal adaptive routing algorithms based on a so-called *pheromone tracking strategy* is proposed in this contribution. Minimizing the size of spanning trees (total multicast communication traffic) will not only reduce the communication energy but also decrease the probability of forming spanning trees having slower data rates.

6.5.1 Routing and Multicasting Procedure

The routing engine (RE) units in XHiNoC consist of combination of a *Routing State Machine* (*RSM*) unit and a *Routing Reservation Table* (*RRT*) unit. The combination is aimed at supporting a *runtime link interconnect configuration*.

Definition 2. A Multicast Routing Slot within a Routing Reservation Table is defined as

$$T_{mcs}(k, r_{dir}) \in \{0, 1\} \tag{6.1}$$

where $k \in \Omega = \{0, 1, 2, \cdots, N_{slot} - 1\}$, N_{slot} is the number of ID slots on a link. r_{dir} is a routing direction, where $r_{dir} \in D = \{1, 2, 3, \cdots, N_{outp}\}$, and N_{outp} is the number of I/O ports in a router.

Definition 3. A Routing Reservation Table (RRT) of the RE unit at an input port of a multicast router is defined as

$$T(k) = [T_{mcs}(k, 1) \;\; T_{mcs}(k, 2) \;\; \cdots \;\; T_{mcs}(k, N_{outp})] \tag{6.2}$$

$T(k)$ contains a binary-element vector. Hence, T has 2D (matrix) size of $row \times column = N_{slot} \times N_{outp}$.

Based on Definitions 2 and 3, a binary-encoded multicast routing direction $r_{dir}^{bin} = enc(r_{dir})$ is introduced and has a size of N_{outp} number of binary elements. For example, if $N_{outp} = 5$, then $enc(1) = [1\ 0\ 0\ 0\ 0]$, $enc(2) = [0\ 1\ 0\ 0\ 0]$, $enc(3) = [0\ 0\ 1\ 0\ 0]$, $enc(4) = [0\ 0\ 0\ 1\ 0]$ and $enc(5) = [0\ 0\ 0\ 0\ 1]$. Algorithm 1 describes the ID-based Routing Organization between the *RSM* and the *RRT* unit. If a *RE* unit identifies a flit $F(type, I)$ as a header flit ($type = header$) from the output of a FIFO buffer with local ID-tag $I \in \Gamma$, where $\Gamma = \Omega$ then the routing function of *RSM* unit $f_{RSM}(A_{dest})$ will determine a routing direction r_{dir}. This means that the r_{dir} is calculated logically based on destination address A_{dest} attached in the header flit and current address of the router, and the routing direction is written in the slot number $k = I$ of the *RRT* unit. In the next time periods, when the *RE* units identify payload flits with the same ID-tag number (ID-tag number I) with the previously forwarded header flit, then their routing direction will be taken up directly from the slot number $k = I$ in the *RRT* unit.

Algorithm 1 Runtime IDMA-based multicast routing mechanism

Read Data Flit from Queue : $F_n(type, I)$

$k \Leftarrow I$; A_{dest} is obtained from Header flits
BEGIN *Multicast routing* (r_{dir}^{bin})
if *type* is *Header* **then**
    $r_{dir} \Leftarrow f_{RSM}(A_{dest})$; $T(k, r_{dir}) \Leftarrow 1$
    $r_{dir}^{bin} = enc(r_{dir})$
else if *type* is *Response* **then**
    $r_{dir} \Leftarrow f_{RSM}(A_{dest})$
    $r_{dir}^{bin} = enc(r_{dir})$
else if *type* is *Databody* **then**
    $r_{dir}^{bin} \Leftarrow T(k)$
else if *type* is *Tail* **then**
    $r_{dir}^{bin} \Leftarrow T(k)$; $T(k) \Leftarrow \emptyset$
end if
END *Multicast routing*

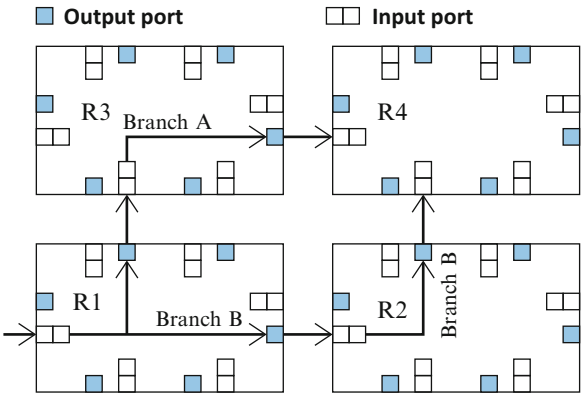

Fig. 6.10 Example of an inefficient spanning tree in adaptive tree-based multicasting

There are three main steps to send a multicast message towards multiple destinations. The first step is to forward all header flits for the multicast tree routing setup and ID-slot reservation. The second step is to multicast (replicate) the payload flits to follow the path set up previously by the header flits. The last step is to set free the reserved local ID-slot by the tail flit. The detail procedure can be found in [24] and is formally described in Algorithm 1.

Definition 4. A runtime tree-based multicast routing configuration of a message that will be sent to a number of N_{dest} multicast destinations is established by sending N_{hf} number of header flits, where $N_{dest} = N_{hf}$. The multicast header flits $H_j(I), j \in \{1,2,\cdots,N_{hf}\}$ can be ordered arbitrarily, where I is the ID-tag of the headers at a certain (input) link n. Thus, we can further define that $F_n(header,I) = H_j(I)$.

6.5.2 *Inefficient Spanning Tree Problem*

The aim of an efficient adaptive multicast routing is to minimize the communication energy. Figure 6.10 shows an example of an inefficient adaptive routing. A tree-based multicast message coming from the WEST input port of the router $R1$ forms two branches in different routing directions i.e., a branch to NORTH (branch A) and a branch to EAST (branch B) direction. It can be assumed, that the branches A and B are established by header flit 1 and header flit 2, belonging to the same multicast message. The header flits will be routed to multicast destinations (x_{t1}, y_{t1}) and (x_{t2}, y_{t2}), respectively, by using a *minimal adaptive routing algorithm*.

Postulate 1. *In a regular 2D mesh network, if a header flit $H_j(I)$ in a current mesh node (x,y) will be routed to a destination node (x_j, y_j) by using a minimal adaptive routing algorithm, then there will be at most two alternative output port directions (otherwise the routing path would not be minimal). When $|x_{offs,j}| = |x_j - x| > 0$ and $|y_{offs,j}| = |y_j - y| > 0$, then there will be two alternative output directions, i.e. m_1 and m_2, where if m_1 is an output direction that can reduce $|x_{offs,j}|$, then m_2 is an output direction that can reduce $|y_{offs,j}|$, or vice versa.*

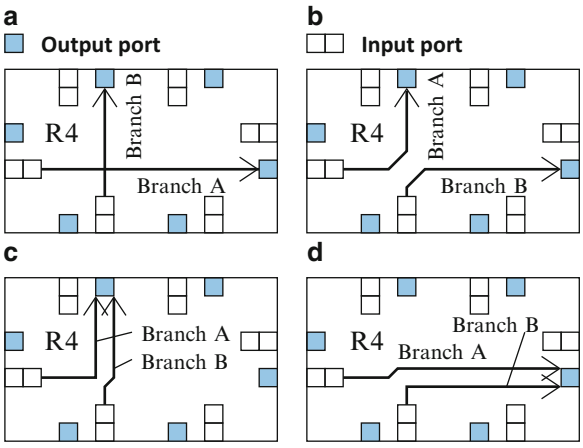

Fig. 6.11 Consequences of the inefficient spanning trees

In the router $R3$ in Fig. 6.10, the multicast message is routed from SOUTH to EAST direction (branch A), while in the router $R2$, the multicast message is routed from WEST to NORTH direction (branch B). Finally these two branches are then routed to the same router (router $R4$). In this case, the multicast tree branches (spanning trees) are inefficient in term of communication energy. The communication energy can be reduced if the router $R1$ performs only the multicast tree branch A or branch B.

Postulate 2. *If two header flits ($H_j(I)$ and $H_k(I)$) having the same ID-tag I (hence, belonging to the same multicast message) are routed from the same input port n in a router node (x,y) at two consecutive times t_{H_j} and t_{H_k} where $t_{H_j} < t_{H_k}$ (which means that H_j is routed firstly before H_k), then an inefficient runtime spanning tree configuration can happen when H_k (which will be routed to destination node (x_k, y_k), where $|x_{offs,k}| = |x_k - x| > 0$ and $|y_{offs,k}| = |y_k - y| > 0$) does not follow to an output port m which has also been selected previously for routing of H_j (having destination node (x_j, y_j)), where $\langle x_j - x = x_{offs,j} = x_{offs,k}$ and $y_j - y = y_{offs,j} = y_{offs,k}\rangle$ or $\langle x_{offs,j} = 0$ and $y_{offs,j} = y_{offs,k}\rangle$ or $\langle x_{offs,j} = x_{offs,k}$ and $y_{offs,j} = 0\rangle$.*

Figure 6.11 depicts four possible situations that can occur in the router $R4$ as the further disadvantageous consequences of the inefficient multicast spanning trees configured from Fig. 6.10. These situations can happen since the number of free ID slots on each communication link as the parameter of the adaptive routing algorithm may change dynamically. Figure 6.11a, b show a *tree-branch crossover problem*, in which the inefficient spanning trees are propagated through different outgoing ports. If we assume that the current address of router $R4$ is (x_{curr}, y_{curr}) and the target nodes of the tree branches A and B are (x_{t1}, y_{t1}) and (x_{t2}, y_{t2}) such that $x_{offset1} = x_{t1} - x_{curr} > 0$ and $y_{offset1} = y_{t1} - y_{curr} > 0$ as well as $x_{offset2} = x_{t2} - x_{curr} > 0$ and $y_{offset2} = y_{t2} - y_{curr} > 0$, then in any circumstance, the inefficient situation might happen again in the next intermediate nodes.

Figure 6.11c, d illustrate a *tree-branch interference problem*, where the inefficient spanning trees interfere into the same outgoing port. This situation will lead to inefficient multicast communication time (increase of communication latency and data workloads) because of the self-contention problem. Two multicast messages will be forwarded from different input ports to the same output port, carrying identical data payload (double workloading).

6.5.3 Solution: Efficient Adaptive Routing Selection with Pheromone Tracking Strategy

The problematic configurations presented in Figs. 6.10 and 6.11 are not only inefficient in terms of communication energy (because the inefficient traffic will overburden the NoC), but also in term of communication latency, since the inefficient traffic can degrade the data rate of the multicast traffic. The problems reduce the NoC performance while increasing power consumption.

We solve the aforementioned problem not by designing a specific multicast path optimization algorithm that should be run at compile time or before injecting multicast messages (*pre-processing algorithm*). Compile-time path optimization algorithms such as the optimal spanning tree algorithm are suitable for *source routing approach*, where the routing paths for the overall pathes of a multicast message from source to destination node are determined at source node before the message is injected to the network. In contrast, in XHiNoC the routing algorithm used to route unicast and multicast messages is the same, the routing decisions are made at runtime and locally executed hop-by-hop on every port of each router. Thus, we do not follow the approach of a path pre-processing optimization algorithm for the sake of initiation-time-efficiency.

In order to avoid the previously described problems, each time a routing engine has two alternative output ports for making a routing decision, then a new adaptive output selection strategy between two alternative output ports will be applied. A simple abstract view of the adaptive selection strategy is outlined in Algorithm 2. The basic concept of the proposed algorithm is the identification of the track records (*pheromone* trails) of other previously-routed header flits that belong to the same multicast message. This concept is designed in order to avoid inefficient spanning trees branches of the multicast tree.

Definition 5. A pheromone trail checking is an operation to check the binary state of the multicast routing slots $T_{mcs}(k,m_1)$ and $T_{mcs}(k,m_2)$. The operation is made by a header flit $H_j(I)$ having ID-tag number $I = k$ that will be alternatively routed to output directions m_1 and m_2, where $m_1, m_2 \in D$.

The hardware implementation of the efficient adaptive routing selection function with pheromone tracking strategy will not result in a more complex operation in the XHiNoC microarchitecture. The router complexity can be reduced naturally

Algorithm 2 Multicast: adaptive routing selection strategy

```
1: Begin Function Select(port m1, port m2)
2: if Routing can be made to port m1 and port m2 then
3:    if Routing for the same Multicast Packet has been made to port m1 and not
      yet to port m2 then
4:       Return Routing = port m1
5:    else if Routing for the same Multicast Packet has been made to port m2 and
      not yet to port m1 then
6:       Return Routing = port m2
7:    else if Routing for the same Multicast Packet has not been made to port m1
      and port m2, or has been made both to port m1 and port m2 then
8:       if UsedID(port m1) < UsedID(port m2) then
9:          Return Routing = port m1
10:      else
11:         Return Routing = port m2
12:      end if
13:   end if
14: end if
15: End Function
```

Algorithm 3 Logical view of Algorithm 2

Incoming Data Flit : $F_n(type, ID)$
$T(k,m)$: The slot of RRT for a flit with ID=k to direction m
$UsedID(m)$: Number of used/reserved ID slots in direction m

```
1: Begin Function Select(m1, m2)
2: k ⇐ ID
3: if ¬T(k,m1) & UsedID(m1) ≤ ¬T(k,m2) & UsedID(m2) then
4:    Return m1
5: else if ¬T(k,m1) & UsedID(m1) > ¬T(k,m2) & UsedID(m2) then
6:    Return m2
7: end if
8: End Function
```

by using the advantageous feature of our dynamic runtime table-based routing management and control. In order to achieve an efficient operation, the logical view of the multicast adaptive routing selection strategy is proposed in Algorithm 3.

The operation of Algorithm 3 (logical view of the efficient adaptive multicast routing algorithm) in accordance with Definition 5 (pheromone trail checking strategy) will be illustrated and explained in the following example. Let us assume that a multicast header having ID-tag $k = 3$ can be alternatively routed to output port m_1 and m_2, where the number of used (already reserved) ID-tags at the output ports is $UsedID(m_1) = 0101(5)$ and $UsedID(m_2) = 0010(2)$, respectively. Let us also assume that a previously routed header, which belongs to the same multicast group as the current header (also with ID-tag $k = 3$), has been routed to port m_1 such that $T(3,m_1) = 1$, and there is no header with ID-tag $k = 3$ that has been routed to port m_2 such that $T(3,m_2) = 0$. Based on

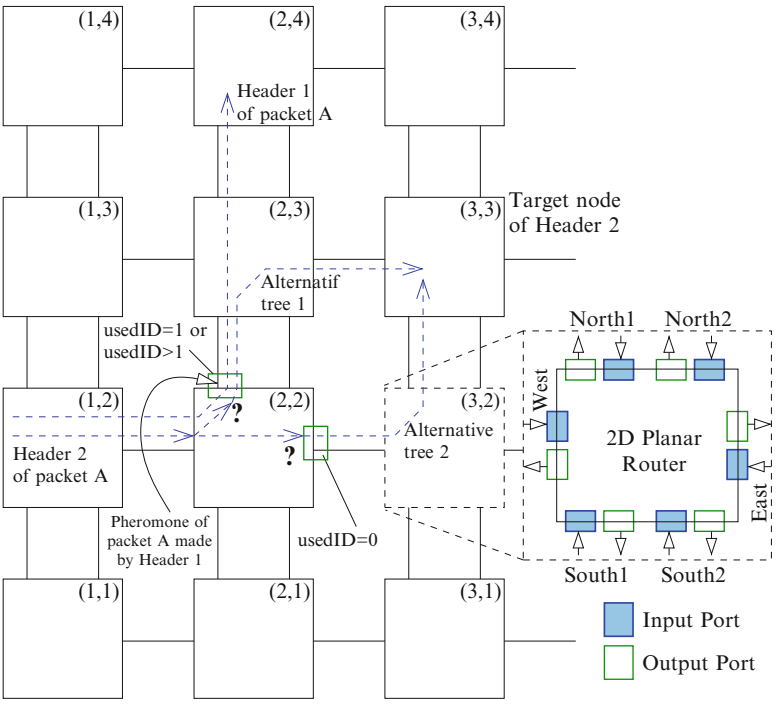

Fig. 6.12 Example illustration of the pheromone tracking strategy

Algorithm 3, we evaluate $\neg T(3,m_1)$ & $UsedID(m_1) < \neg T(3,m_2)$ & $UsedID(m_2)$, or 0 & 0101 (00101) $<$ 1 & 0010 (10010). The operator "¬" is a negation operator. The operator "&" represents a concatenation operator, such that the negation of $T(k,m)$ is inserted as the most-significant-bit of the $UsedID(m)$ logical value. Therefore, according to Algorithm 3. the header is then routed to port m_1, because it has lower cost value than the port m_2.

For the sake of clarity, Fig. 6.12 illustrates how a packet header will track a pheromone made by other header flit that belong to the same packet, in order to set up an efficient spanning tree. As shown in this figure, Header 1 of packet A has arrived router node (2,4). The path traversed by Header 1 is shown in the figure. At the same time, Header 2 of packet A is now at router node (2,2), targeting the destination router node (3,3). So Header 2 can select either output port East or North 1 to configure one of two alternative spanning tree branches as depicted in the figure. The number of used IDs at output port North 1 is 1 or probably more than 1 if any other packets have also reserved IDs. Although the number of used IDs at East direction outport is 0 or less than the number of used IDs at output port North 1, Header 2 will finally select the North 1 port to configure the alternative spanning tree branch 1, because Header 2 has detected a pheromone trail made by Header 1, which has been previously routed at the router node (2,2).

According to Definition 5 and Algorithm 3, Header 2 can detect the pheromone of Header 1 and track/follow the same spanning tree made by the Header 1, because both header flits belong to the same packet, i.e. both have the same local ID-tag at each communication link. From Fig. 6.12, we can see that by following the pheromone tracking strategy, Header 2 will finally add only one communication link (link between nodes (2,3)–(3,3), through alternative tree 1), instead of two communication links (links between nodes (2,2)–(3,2) and between node (3,2)–(3,3), through alternative tree 2). Therefore, the communication energy of the multicast communication performed by the packet A will be less and the overall communication more efficient.

It it is obvious, that the pheromone trail tracking can be efficiently and logically implemented by detecting the bit-value of the routing slot $T(k,m)$. This detected bit is used as the most-significant bit (MSB) in the concatenation operation with the $UsedID$ bit-signal, i.e. $T(k,m)\&UsedID(m)$. In general, the router will route a header flit into an output port having more free ID-tags. However, the pheromone trail, which is represented by the routing slot $T(k,m)$, has a higher priority over the number of already reserved ID slots. This pheromone trail checking method is used to avoid inefficient multicast spanning tree as presented visually in Fig. 6.10.

In this section four algorithms that are used to route packets in NoCs have been presented. Algorithm 1 is an ID-based routing algorithm that is generally used in the XHiNoC router by default. Algorithm 4 represents a planar adaptive routing scheme, that routes packets in the 2D planar NoC using two sub-networks with dual physical channels in the north and south direction. The most important contributing algorithms are Algorithms 2 and 3. These algorithms are solving the problem of the inefficient adaptive tree-based multicast. The algorithms can also be simply derived and implemented based on the existing physical state information of the router microarchitecture.

6.6 Experimental Results

In this section, four XHiNoC multicast router prototypes with different multicast routing algorithms are compared. The first prototype is a multicast router working with a planar adaptive routing algorithm on a NoC mesh architecture, which is presented with **'plnr'** acronym in the figures. The second prototype applies a static XY multicast routing algorithm on the mesh standard architecture (**'xy'**). The third and fourth prototype are multicast routers in a standard mesh architecture applying an adaptive West-First (WF) routing algorithm (**'wf-v1'** and **'wf-v2'**). The adaptive WF multicast router version 1 (**'wf-v1'**) is a multicast router without implementation of an adaptive selection strategy to avoid an inefficient spanning tree (branches of the multicast tree) as presented in [24]. Thus in this prototype, the multicast trees are formed freely without considering the track records of the other previously-routed header flits belonging to the same multicast group. The adaptive WF multicast router version 2 (**'wf-v2'**) implements the adaptive selection strategy presented in the Algorithm 2 to avoid an inefficient spanning tree problem.

Algorithm 4 2D Planar adaptive routing algorithm

Network is partitioned into two subnets: X^+ and X^- Subnet.
Set of output ports in X^+ Subnet: $\{EAST, SOUTH\ 1, NORTH\ 1, LOCAL\}$.
Set of output ports in X^- Subnet: $\{WEST, SOUTH\ 2, NORTH\ 2, LOCAL\}$.
Select(m_1, m_2) is selection function between output port m_1 or m_2.

1: $X_{offs} = X_{target} - X_{source}$
2: $Y_{offs} = Y_{target} - Y_{source}$
3: **while** *Packet* is in *Subnet* X^+ i.e.$(X_{offs} \geq 0)$ **do**
4: **if** $X_{offs} = 0$ and $Y_{offs} = 0$ **then**
5: $Routing = LOCAL$
6: **else if** $X_{offs} = 0$ and $Y_{offs} > 0$ **then**
7: $Routing = NORTH\ 1$
8: **else if** $X_{offs} = 0$ and $Y_{offs} < 0$ **then**
9: $Routing = SOUTH\ 1$
10: **else if** $X_{offs} > 0$ and $Y_{offs} = 0$ **then**
11: $Routing = EAST$
12: **else if** $X_{offs} > 0$ and $Y_{offs} > 0$ **then**
13: $Routing$=**Select**$(NORTH\ 1, EAST)$
14: **else if** $X_{offs} > 0$ and $Y_{offs} < 0$ **then**
15: $Routing$=**Select**$(SOUTH\ 1, EAST)$
16: **end if**
17: **end while**
18: **while** *Packet* is in *Subnet* X^- i.e.$(X_{offs} \leq 0)$ **do**
19: **if** $X_{offs} = 0$ and $Y_{offs} = 0$ **then**
20: $Routing = LOCAL$
21: **else if** $X_{offs} = 0$ and $Y_{offs} > 0$ **then**
22: $Routing = NORTH\ 2$
23: **else if** $X_{offs} = 0$ and $Y_{offs} < 0$ **then**
24: $Routing = SOUTH\ 2$
25: **else if** $X_{offs} < 0$ and $Y_{offs} = 0$ **then**
26: $Routing = WEST$
27: **else if** $X_{offs} < 0$ and $Y_{offs} > 0$ **then**
28: $Routing$=**Select**$(NORTH\ 2, WEST)$
29: **else if** $X_{offs} < 0$ and $Y_{offs} < 0$ **then**
30: $Routing$=**Select**$(SOUTH\ 2, WEST)$
31: **end if**
32: **end while**

The experiment is setup by applying a multicast random data distribution (traffic) scenario to validate the theorem and methodology of the proposed deadlock-free multicast routing. Figure 6.13 depicts the distribution of the source-destination unicast-multicast communication partners in the 2D 8×8 mesh architecture (64 network nodes). The numerical symbol at the bottom and the left side of the mesh network represents the 2D node x-address and y-address. From the figure it can be seen, that 9 multicast communication partners (**M6**, **M8**) and 4 unicast communication pairs (**U**) are placed. Three of 9 multicast communication sources have 8 multicast targets (**M8**), while the remaining 6 multicast sources have 6 multicast destinations (**M6**).

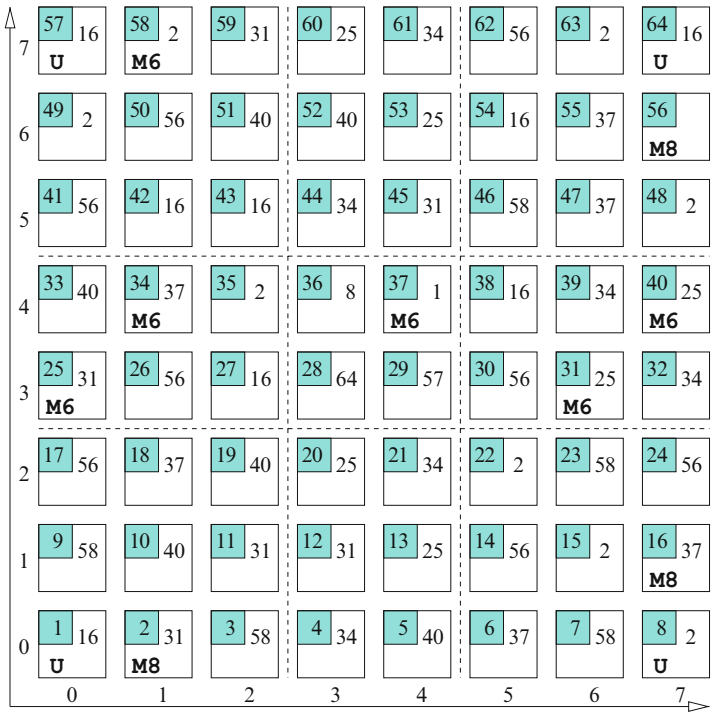

Fig. 6.13 Distribution of the source-destination communication partners

Every NoC router in Fig. 6.13 is depicted with a square block with numerical attributes. A numerical symbol in the small square block at the top-left side of a NoC router node represents the node number. The numerical symbol at the top-right side in the NoC router node represents the communication partner of the node, from which the NoC router node will receive a message. For example, the network node at node address $(2,1)$ (2D node address, 2 is the x-horizontal address and 1 is the y-vertical address) has node number 11. At the top-right side in this node, we see the numerical value 31 (this means that the node will receive a packet from mesh node 31 located in the node address $(6,3)$).

The boldface symbols (**U**, **M6** and **M8**) at the bottom-left corner of the router box represent that the network node will send a unicast message (**U**) or a multicast message with a number of 6 target nodes (**M6**) or 8 target nodes (**M8**). For example, the mesh node at address $(7,1)$ (mesh node number 16) is attributed with (**M8**). This means that the node will send a multicast message into 8 destination nodes. We can find the target nodes of the multicast message sent from the mesh node number 16 by identifying mesh nodes having numerical symbol 16 at the right-side in the router box. In order to find easily the partners of each unicast and multicast communications, Table 6.1 gives an overview on the unicast and multicast communication partners/groups of the source-destination distribution presented in Fig. 6.13.

Table 6.1 Unicast and multicast communications for the random multicast test traffic scenario

Comm. group	Type	Source	Targ. 1	Targ. 2	Targ. 3	Targ. 4	Targ. 5	Targ. 6	Targ. 7	Targ. 8
Comm. 1	M6	(1,4)	(6,4)	(7,3)	(4,2)	(3,0)	(3,5)	(4,7)	–	–
Comm. 2	M6	(7,4)	(0,4)	(3,6)	(2,6)	(2,2)	(1,1)	(4,0)	–	–
Comm. 3	M6	(0,3)	(6,3)	(3,2)	(4,1)	(7,4)	(4,6)	(3,7)	–	–
Comm. 4	M6	(6,3)	(0,3)	(4,5)	(3,1)	(2,1)	(1,0)	(2,7)	–	–
Comm. 5	M6	(1,7)	(7,6)	(5,5)	(6,2)	(6,0)	(2,0)	(0,1)	–	–
Comm. 6	M6	(4,4)	(1,4)	(1,2)	(7,1)	(5,0)	(6,5)	(6,6)	–	–
Comm. 7	M8	(1,0)	(7,0)	(6,1)	(5,2)	(2,4)	(7,5)	(6,7)	(0,6)	(1,7)
Comm. 8	M8	(7,1)	(2,3)	(5,4)	(2,5)	(1,5)	(5,6)	(0,7)	(7,7)	(0,0)
Comm. 9	M8	(7,6)	(1,6)	(0,5)	(5,3)	(1,3)	(0,2)	(5,1)	(5,7)	(7,2)
Comm. 10	U	(0,0)	(4,4)	–	–	–	–	–	–	–
Comm. 11	U	(0,7)	(4,3)	–	–	–	–	–	–	–
Comm. 12	U	(7,0)	(3,4)	–	–	–	–	–	–	–
Comm. 13	U	(7,7)	(3,3)	–	–	–	–	–	–	–

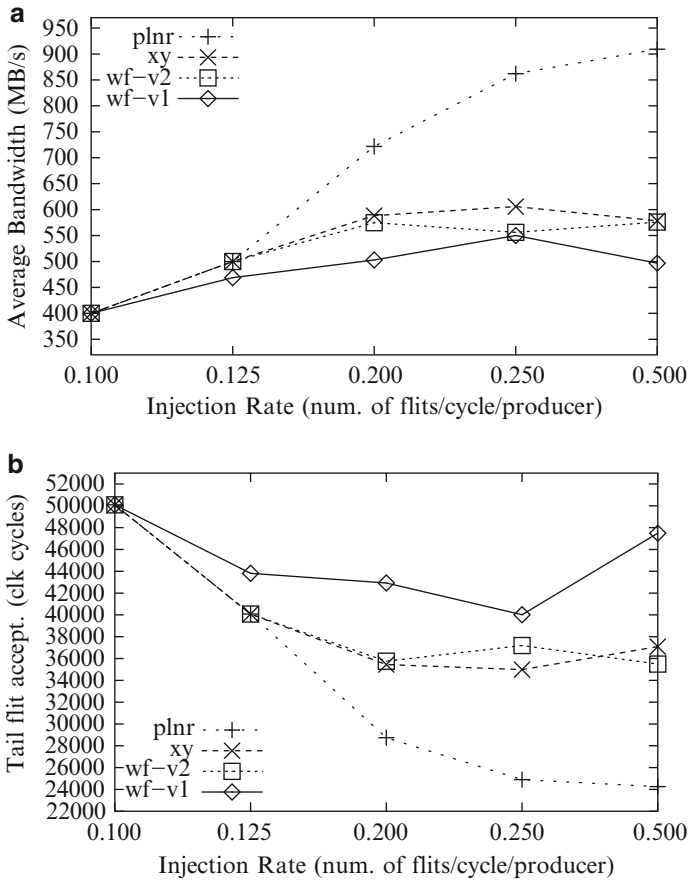

Fig. 6.14 Average bandwidth and tail flit arrival latency measurement versus expected data injection rates for multicast random test scenario. (**a**) Average bandwidth. (**b**) Average tail flit latency

6.6.1 Performance Measurements

The measurements of the average bandwidth and tail flit acceptance latency with various expected data injection rates are depicted in Fig. 6.14. The measurements are made for five different expected data injection rates, i.e. 0.1, 0.125, 0.2, 0.25 and 0.5 flits/cycle (fpc) where each source node injects a 5000-flit packet (equivalent with $4 \times 5{,}000 = 20$ kB data words). It seems that for all multicast routing algorithms, the average BW increases as the injection rate is increased. However, the average BW will tend to saturate, when the injection rate is set nearly to maximum injection rate (the maximum injection rate is 1 flit/cycle).

Table 6.2 Total performed traffic on each link direction for different tree-based static and adaptive multicast routing methods

Routers	South2	North2	South	West	North	East	TOT.
plnr	30	28	34	65	27	46	230
xy	–	–	83	40	89	36	248
wf-v2	–	–	79	40	80	46	245
wf-v1	–	–	85	40	81	86	292

Based on measurements with 1 GHz data frequency, each link has a maximum bandwidth (BW) capacity of 2,000 MB/s. The number of clock cycles required to transfer the 20 kB data words to a target node j is measured by counting the number of clock cycles until receiving the tail flit of the packet (i.e. the 5,000th or the last flit), which is defined as N_{TC}^{j}. Since there are 64 target nodes in the traffic scenario shown in Fig. 6.13, the average tail flit acceptance values plotted in Fig. 6.14b are $\frac{1}{64}\sum_{j=1}^{64} N_{TC}^{j}$. The BW measurement on a target node j is calculated as $B_j = (N_{TC}^{j})^{-1} \times 20,000$ B. Consequently, the average actual BW values plotted in Fig. 6.14a are $\frac{1}{64}\sum_{j=1}^{64} B_j$. It can be seen, that the tree-based multicast router with planar adaptive routing algorithm shows the best performance both in terms of the average bandwidth (Fig. 6.14a) and the average latency of the tail flits acceptance (Fig. 6.14b) for all expected data rates. If the data injection rates are very low (e.g. 0.1 fpc), then the performance of the multicast routers will be the same. The performance of the planar adaptive multicast router will be significantly better compared to the other multicast routers, when the data is expected to be transmitted with higher data rates.

6.6.2 Communication Energy Evaluation

Table 6.2 shows the comparisons of the total performed communication traffic for the four tree-based multicast router prototype scenarios. The number of the traffic represents the number of communication resources (communication links) being used to route the unicast/multicast message from source to destination nodes. The metric of the traffic number is measured by counting the number of used/reserved ID slots at all router output ports at peak performance. This performance metric (the number of traffic) can also be used as measurement unit for the communication energy of the evaluated multicast routers. This metric is also interesting for data comparison with other works in the future, since it is technology-independent.

As shown in Table 6.2, it seems that the planar adaptive multicast router ('plnr') consumes less communication resources compared to the other multicast routers, i.e. about 230 communication links followed by the west-first adaptive multicast routing with the efficient spanning tree method ('wf-v2'), and then the static tree-based multicast router that uses XY routing algorithm ('xy').

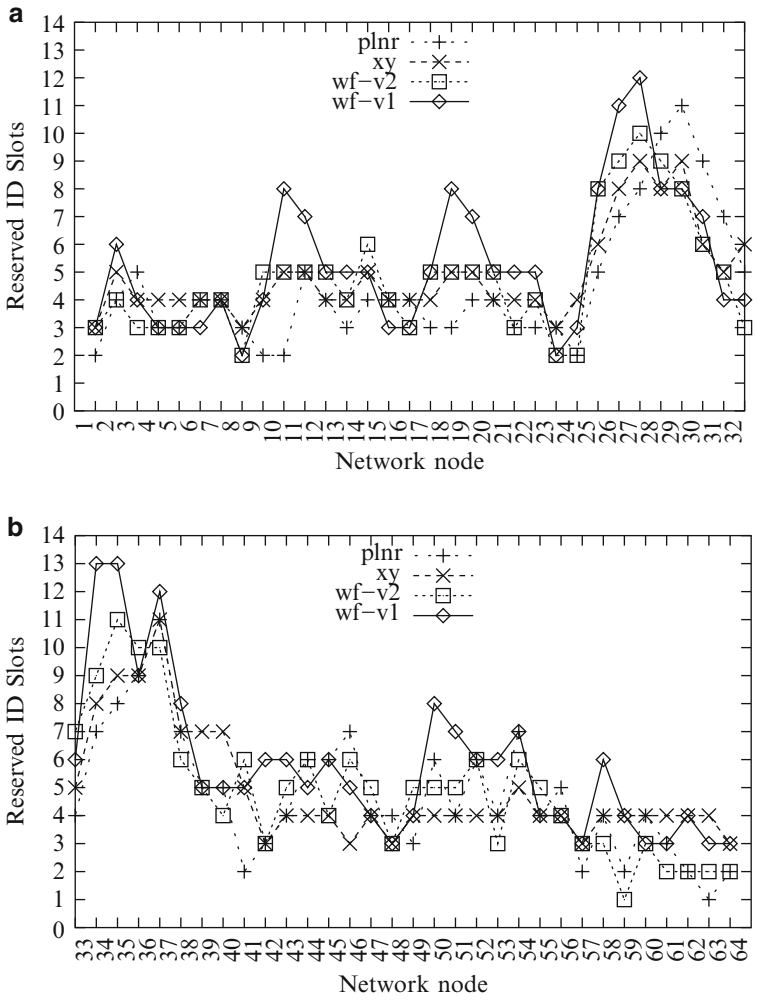

Fig. 6.15 Reserved (used) overall ID slots for multicast random test scenarios. (**a**) Node 1–32. (**b**) Node 32–64

In order to see the traffic configurations in the network in detail, Fig. 6.15 shows the 2D view of the total ID slot reservations in every NoC router node. Figure 6.15a depicts the total reserved ID slots for the NoC router node 1 until node 32, while Fig. 6.15a shows the total ID slots reservation for the NoC router node 33 until node 64. As depicted in Fig. 6.15, the west-first adaptive routing algorithm without the efficient spanning tree method ('wf-v1') reserves more ID-slots than the other routing algorithms at several router nodes.

6.7 State-of-the-Art of Multicast Routing Techniques for NoCs

One basic class of methods for routing multicast messages in mesh-based NoCs are *tree-based multicast routing* techniques. In tree-based multicast routing, a header ordering before submission of the packet within the source node is not required (the order of the destination addresses can be freely determined). The multicast routing will form communication paths like branches of trees connecting the source node with the destination nodes at the end points of the tree branches. The work in [4,19] and [14] have presented the concept and methodology to route multicast messages by using tree-based methods, which has been utilized in general internetworking context. The work in [25] has presented a new theory for deadlock free tree-based multicast routing for networks-on-chip area (mesh topology). The theory is developed based on a dynamic local ID-tag routing organization and the concept of a *hold-release tagging mechanism*.

Alternatively, multicast messages can also be routed by applying path-based multicast routing methods. Here, as a prerequisite, a multiple target ordering is required before the multicast packets are sent to the network. The path-based multicast routing requires a full implementation of an adaptive routing algorithm allowing all turns in the mesh-based network topology. Therefore, virtual channels are usually needed to make a deadlock free multicast routing function. Virtual channels in the context of on-chip interconnection network will consume not only larger logic gate area but also larger power dissipation. The works in [9, 15, 16] and [5] have presented the *path-based multicast routing* methods for a mesh-based network topology.

In the Network-on-Chip (NoC) research area, some multicast NoC architectures have been introduced. Most of them use the tree-based multicast routing method [1, 11, 12, 28], and path-based multicast routing method [8, 13, 18, 21]. The virtual circuit tree multicast (VCTM) NoC [12] for example has presented a NoC that uses virtual circuit tree numbers to configure routing paths. However, compared to the presented approach, which uses runtime dynamic local ID configuration, the VCTM NoC applies a static method, where virtual circuit tables are statically partitioned among nodes.

A few NoC environments also proposed specific multicast routing methodologies such as a closed-loop path routing method [17] and a region-based routing method [21]. The recursive partitioning multicast (RPM) NoC [28], as another example, applies a recursive hop-by-hop network partitioning method to multicast packets at each intermediate node. The packets in the RPM NoC make replication at a certain node to multicast packets. The replicated packets will update the destination list attached on their header flit and make a new network partitioning recursively based on their current position. By using such scheme, the RPM method will increase the complexity of the routing computational logic.

Table 6.3 presents several NoCs that propose and provide multicast routing services for packet routing. Most of the NoC approaches route the network packets

Table 6.3 State-of-the-arts of multicast routing techniques for NoCs

	Multicast method	Switching method	Routing adaptivity	VC buffers, (buffer depth)	Logic area (technology)	Specific features
VCTM [12]	Tree-based	Circuit switching	Adaptive, static	ja. 4,8 (d.n.a)	0.0240[d] mm^2 (70 nm)	Virtual circuit tree, static VC-table partitioning
RPM [28]	Tree-based	Wormhole	d.o. target distribution	ja. 4 (4)	4.0[e] mm^2 (65 nm)	Recursive partitioning, priority-based replication
LDPM [8]	Path-based	Wormhole	Odd-even adaptive	ja[a]. 4 (8)	21,050 gates (0.25 µm)	Path-based with optimized destination ordering
bLBDR [21]	Region-based	Wormhole	Static, adaptive ext.	d.n.a (d.n.a)	0.0499 mm^2 (90 nm)	Traffic isolations, network domain partitioning
MRR [1]	Tree-based	Virtual cut-through	Adaptive	no. (20,20,10)[b]	d.n.a	Extra internal ring buffers (rotary router)
OPT, LXYROPT [11]	Tree-based	Circuit	Adaptive west-first	ja. 4 (3)	d.n.a	Pre-processing algorithm for tree generation
VC-A/D-FD [13]	Path-based	Wormhole, VCT	Static, adaptive	ja. 4 (8/10)[c]	1,172.03[f] $M\lambda^2$ (65 nm)	FIFO with address/data decoupling
Custom Mcast [29]	Tree-based	Packet	Static	no. (4)	0.18–3.06 mm^2 (70 nm)	Multicast routing at design-time (static)
COMC [18]	Path-based	Wormhole	Static	ja. 4, 6 (2)	d.n.a	Connection-oriented path-based multicast
TDM-VCC [17]	Closed-loop path	Circuit	Static	no. (d.n.a)	d.n.a	Pre-processing algorithm for TDM circuit configuration
XHiNoC	Tree-based	Wormhole cut-through	Adaptive 2-net planar	no. (2)	0.1378 mm^2 (130 nm)	Runtime dynamic local ID management

d.n.a. detail not available, *d.o.* depend on, *ext.* extendable, *VC* virtual channel, *VCT* virtual cut-through

[a]Implemented as delivery channel buffers to avoid multicast deadlock

[b]20 phits in buffering segment stage, 20 in output stage, 10 in input stage

[c]VC length is 8 for address, 10 for data

[d]Only the table (512 entries), not the overall logic area of the router

[e]One tile area (inclusive tile processor)

[f]The area is for adaptive router with wormhole switching (no further explanation about the λ unit)

by using wormhole switching method. The table compares some aspects regarding router implementation and their specific features. Some information cannot be provided in the table, because the data is not available from the considered publications in the bibliography. As shown in the table, compared to the other NoCs, the XHiNoC can be implemented with a very small size buffer (single buffer with only two data slots per input port). The FIFO Buffer is a NoC component that can consume relatively large logic area compared to other NoC components. It can also have a large power dissipation due to intensive switching activities of data that occurs in the buffer.

In Multiprocessor Systems-on-Chip (MPSoC) and chip-level multiprocessor system (CMP) applications, multicast communication services are an important and essential issue. Recent works related to NoC-based multicast communication are presented in [29] and [17]. The work in [29] presents the problem of synthesizing custom NoC architectures that are optimized for a given application (which is critical, when dependability aspects are considered as well). The there presented multicast method considers both unicast and multicast traffic flow in the input specification. But the work proposes a static solution for deadlock-free multicast routing that is fixed to specific NoC applications, i.e. the applications must be known before chip fabrication. The work presented in [17] proposes a TDM (Time Division Multiplex)-based virtual circuit configuration (TDM-VCC) where a pre-processing algorithm for time slot allocations is made before injecting multicast messages into the NoC. In some specific embedded system applications, the inter-core communication patterns are known. Therefore, a pre-processing static routing for congestion avoiding techniques can be used [20, 29], expectedly resulting in a much simpler router architecture (pre-manufacturing routing technique). Runtime dynamic adaptive routing methods [2, 3, 22] are however an interesting approach in the NoC-based multicore embedded systems, where applications may not be known in advance and dependability and reconfiguration plays a role. Indeed, some embedded IC vendors in the multicore era could potentially not only market IP cores but also system architectures [7], where many applications can be mapped onto the system architectures product (IP+NoC cores). Therefore, the implementation of the runtime dynamic adaptive tree-based multicast routing will simplify an embedded system design flow because the routing information configuration is not needed anymore on the post-manufacture (on-chip) router. In this context however, the runtime techniques will need extra area cost and complexity.

6.8 Summary

In general, NoC routers applying planar adaptive routing schemes can achieve an improved performance because of the higher bandwidth capacity of the NoC in double vertical links connecting NORTH and SOUTH ports. Nevertheless, this performance gain must be paid by logic and routing area overhead to implement the mesh planar router architecture.

The tree-based multicast routing presented in this chapter belongs to the class of runtime distributed routing techniques, in which routing decisions are made locally during application execution time (runtime) on every router/switch node based on a header's destination address. The advantage of this method is, that it scales well with increasing NoC sizes. Hence, the presented technique to prevent the inefficient spanning tree problems has been proven to be feasible and deliver good results. Compared to static methods it will probably result in a *suboptimal* or *near-optimal* multicast spanning tree, but in certain cases, a *global optimal* spanning tree may be attained.

When a static tree-based multicast routing would be used, then the configured multicast spanning trees will always be the similar, although the order of the header probes is changed. For a fixed traffic scenario, the global optimal multicast spanning tree could be attained by finding an optimum ordering of the header flits in one multicast message. The optimum ordering is strongly dependent on the multicast traffic patterns. Such a procedure would require the computation of an optimum ordering algorithm before injecting multicast packets at source nodes. However, the supplement effort will lead to extra computational power and delay, due to the extra pre-processing at the source nodes. Furthermore, the result can also be sub-optimal, since the traffic situation in the network can vary.

This issue has exactly been addressed within the presented approach, which considers local traffic situations dynamically and minimizes the communication resource and energy usage. When the presented adaptive routing algorithms are used, then the spanning tree is formed independently at runtime by header probing and additional consideration of the local traffic situation. The spanning tree topology can vary with the order of header flits in the multicast message and with indeterministic dynamically varying traffic loads in the network.

References

1. P. Abad, V. Puente, J.-A. Gregorio, MRR: enabling fully adaptive multicast routing for CMP interconnection networks, in *Proceedings of the 15th IEEE International Symposium on High Performance Computer Architecture (HPCA 2009)*, Shanghai, 2009, pp. 355–366
2. M.A. Al Faruque, T. Ebi, J. Henkel, Run-time adaptive on-chip communication scheme, in *Proceedings of the 2007 IEEE/ACM International Conference on Computer-Aided Design (ICCAD'07)*, San Jose (IEEE Press, Piscataway, 2007), pp. 26–31
3. G. Ascia, V. Catania, M. Palesi, D. Patti, Implementation and analysis of a new selection strategy for adaptive routing in networks-on-chip. IEEE Trans. Comput. **57**(6), 809–820 (2008)
4. M. Barnett, D.G. Payne, R.A. van de Geijn, J. Watts, Broadcasting on meshes with worm-hole routing. J. Parallel Distrib. Comput. **35**(2), 111–122 (1996)
5. R.V. Boppana, S. Chalasani, C.S. Raghavendra, Resource deadlocks and performance of wormhole multicast routing algorithms. IEEE Trans. Parallel Distrib. Syst. **9**(6), 535–549 (1998)
6. A.A. Chien, J.H. Kim, Planar adaptive routing: low-cost adaptive networks for multiprocessors, in *Proceedings of the 19th International Symposium on Computer Architecture*, Gold Coast, May 1992, pp. 268–277

7. M. Coppola, M.D. Grammatikakis, R. Locatelli, G. Maruccia, L. Pieralisi, *Design of Cost-Efficient Interconnect Processing Units: Spidergon STNoC* (CRC/Taylor & Francis, Boca Raton, 2009)
8. M. Daneshtalab, M. Ebrahimi, S. Mohammadi, A. Afzali-Kusha, Low-distance path-based multicast routing algorithm for network-on-chips. IET Comput. Digit. Tech. **3**(5), 430–442 (2009)
9. J. Duato, A theory of deadlock-free adaptive multicast routing in wormhole networks. IEEE Trans. Parallel Distrib. Syst. **6**(9), 976–987 (1995)
10. J. Duato, S. Yalamanchili, L. Ni, *Interconnection Networks: An Engineering Approach*, Revised Printing (Morgan Kaufmann, San Francisco, 2003)
11. W. Hu, Z. Lu, A. Jantsch, H. Liu, Power-Efficient tree-based multicast support for networks-on-chip, in *Proceedings of the 16th Asia and South Pacific Design Automation Conference (ASP-DAC'11)*, Yokohama, 2011, pp. 363–368
12. N.E. Jerger, L.S. Peh, M. Lipasti, Virtual circuit tree multicasting: a case for on-chip hardware multicast support, in *Proceedings of the 35th Annual Internationl Symposium on Computer Architecture (ISCA'08)*, Beijing, 2008, pp. 229–240
13. K.-M. Keung, A. Tyagi, Breaking adaptive multicast deadlock by virtual channel address/data FIFO decoupling, in *Proceedings of the 2nd International Workshop on Network-on-Chip Architecture (NoCArch'09)*, New York, 2009, pp. 11–16
14. D.R. Kumar, W.A. Najjar, P.K. Srimani, A new adaptive hardware tree-based multicast routing in K-Ary N-cubes. IEEE Trans. Comput. **50**(7), 647–659 (2001)
15. X. Lin, L.M. Ni, Multicast communication in multicomputer networks. IEEE Trans. Parallel Distrib. Syst. **4**(10), 1105–1117 (1993)
16. X. Lin, P.K. McKinley, L.M. Ni, Deadlock-free multicast wormhole routing in 2-D mesh multicomputers. IEEE Trans. Parallel Distrib. Syst. **5**(8), 793–804 (1994)
17. Z. Lu, A. Jantsch, TDM virtual-circuit configuration for network-on-chip. Trans. Very Larg. Scale Integr. Syst. **16**(8), 1021–1034 (2008)
18. Z. Lu, B. Yi, A. Jantsch, Connection-oriented multicasting in wormhole-switched network-on-chip, in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI'06)*, Karlsruhe, 2006, pp. 205–210
19. M.P. Malumbres, J. Duato, J. Torrelas, An efficient implementation of tree-based multicast routing for distributed shared-memory multiprocessors, in *Proceedings of the 8th IEEE Symposium on Parallel and Distributed Processing*, New Orleans, 1996, pp. 186–189
20. M. Palesi, R. Holsmark, S. Kumar, V. Catania, Application specific routing algorithms for networks on chip. IEEE Trans. Parallel Distrib. Syst. **20**(3), 316–330 (2009)
21. S. Rodrigo, J. Flich, J. Duato, M. Hummel, Efficient unicast and multicast support for CMPs, in *Proceedings of the 41st IEEE/ACM International Symposium on Microarchitecture (MICRO-41)*, 2008, pp. 364–375
22. F.A. Samman, T. Hollstein, M. Glesner, Multicast parallel pipeline router architecture for network-on-chip, in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'08)*, Munich, Mar 2008, pp. 1396–1401
23. F.A. Samman, T. Hollstein, M. Glesner, Networks-on-Chip based on dynamic wormhole packet identity management. VLSI Des. J. Hindawi Publ. Corp. **2009**, 1–15 (2009)
24. F.A. Samman, T. Hollstein, M. Glesner, Adaptive and deadlock-free tree-based multicast routing for networks-on-chip. IEEE Trans. Very Larg. Scale Integr. Syst. **18**(7), 1067–1080 (2010)
25. F.A. Samman, T. Hollstein, M. Glesner, New theory for deadlock-free multicast routing in wormhole-switched virtual-channelless networks-on-chip. IEEE Trans. Parallel Distrib. Syst. **22**(4), 544–557 (2011)
26. F.A. Samman, T. Hollstein, M. Glesner, Wormhole cut-through switching: flit-level messages interleaving for virtual-channelless network-on-chip. Elsevier Sci. J. Microprocess. Microsyst. Embed. Hardw. Des. **35**(3), 343–358 (2011)

27. F.A. Samman, T. Hollstein, M. Glesner, Planar adaptive network-on-chip supporting deadlock-free and efficient tree-based multicast routing method. Elsevier Sci. J. Microprocess. Microsyst. Embed. Hardw. Des. **36**(6), 449–461 (2012)
28. L. Wang, Y. Jin, H. Kim, E.J. Kim, Recursive partitioning multicast: a bandwidth-efficient routing for networks-on-chip, in *Proceedings of the 3rd ACM/IEEE International Symposium on Networks-on-Chip (NOCS'09)*, San Diego, 2009, pp. 64–73
29. S. Yan, B. Lin, Custom networks-on-chip architectures with multicast routing. Trans. Very Larg. Scale Integr. Syst. **17**(3), 342–355 (2009)
30. H. Zimmermann, OSI reference model – the ISO model of architecture for open systems interconnection. IEEE Trans. Commun. **8**(4), 425–432 (1980)